UNIVERSITAT POLITÈCNIA DE CATALUNYA - ESCOLA SUPERIOR D'ENGINYERIES INDUSTRIAL, AEROESPACIAL I AUDIOVISUAL DE TERRASSA

Project of designing and implementing the orientation control system of a wind turbine model

Author:
Carlos Espinós Garcia

Director:
Joseba Quevedo Casin

Bachelor's Degree in Aerospace Technology Engineering

Delivery date: 22/06/2016

Document: **Technical Sheets**

Contents

LIST OF FIGURES

List of Figures

1 Components tests

The different components that were tested to make sure they could work for the project are listed below.

- Fan's motor: it should be checked that it has enough power to make the wind turbine's blade move.
- Wind turbine's rotary encoder: it should be proved that it can measure the angular rotation of the wind turbine's hub appropriately.
- Wind turbine's motor: it should be tested to ensure that it has enough torque to rotate wind turbine's hub.
- Wind vane rotary encoder: it should be checked that the wind direction is well measured.

1.1 Fan's motor

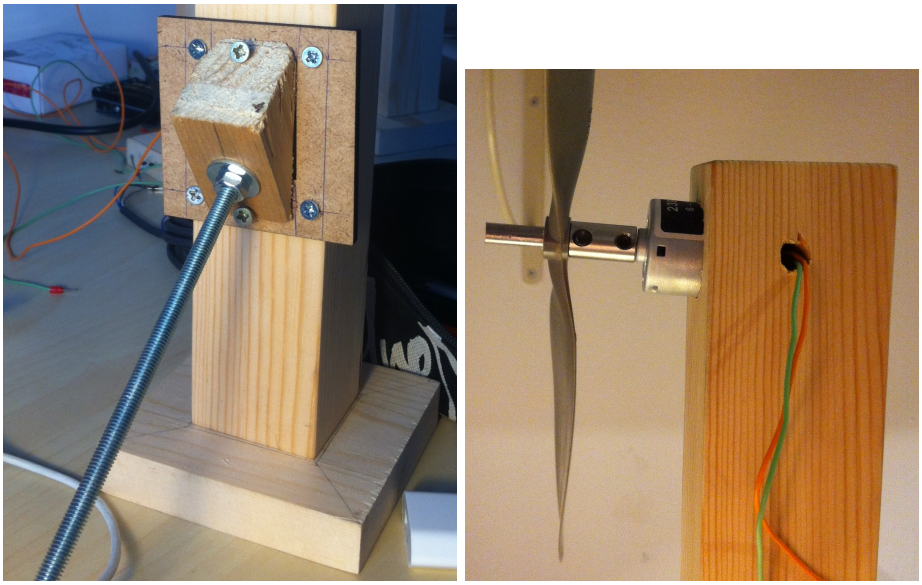
The first step in order to test this component is to assemble all the sub-components. Then, the connections between electronics devices should be made. And, finally, the code used to perform the test is shown.

1.1.1 Assembly

Some instructions are given below in order to accomplish this task. The final result is shown in Figures 1 and 2.

1. Motor [1].
    - Solder the two motor terminals to some wires. It is better that the wires are of different colours in order to distinguish them when connecting to the motor driver. If they are connected in an opposite way, the motor will spin in the opposite sense, thus providing thrust in the wrong direction.
    - Make sure the tower has a hole for the wires. As the motor will be built-in to it, the wires need to pass through this hole to connect with the motor driver.
    - Introduce the motor into the tower. If necessary the whole for the motor could be filed it it is too small. However, note that the motor should be built-in to the tower, so some pressure is always needed to introduce it.
2. Propeller [2].
    - Cut the fan's shaft with a saw. A 3 cm long shaft is enough. Note that the bar used to make the shaft should be plain. If it is threaded, the propeller's rotation could loosen it.
    - Attach the propeller to the shaft. This link should be a built-in one. Then, it is required to apply some force on the shaft to if it into the propeller's hole. If it is bigger than the shaft, it can be recovered with some paper or teflon.
3. Couple the motor's shaft with the propeller's shaft using the shaft coupler.

4. It was observed that the motor was so powerful that an auxiliary structure was necessary to avoid the "fan" tower from rotating back.
    - Cut two pieces of wood in the same triangular form. The triangle should have a 90º angle. A 60º, 30º and 90º triangle with a 6 cm long hypotenuse is proposed. It should be about 3 cm thick.
    - Drill a hole for a threaded bar of 6 mm diameter in the two pieces. One hole should be made on the triangle's hypotenuse while the other should be made on the largest cathetus. In this way, the threaded bar will form a 30º angle with the ground, if the proposed shapes are used.
    - The piece with a hole in the hypotenuse should be attached to a wood plate (not thicker than 0,5 cm) thanks to two screws. The screw hole can be made with a gimlet.
    - Join the wood plate to the fan's tower through 4 screws. Their hole can also be made thanks to a gimlet.
    - Fasten the threaded bar to the wood pieces with a pair of washers and nuts.



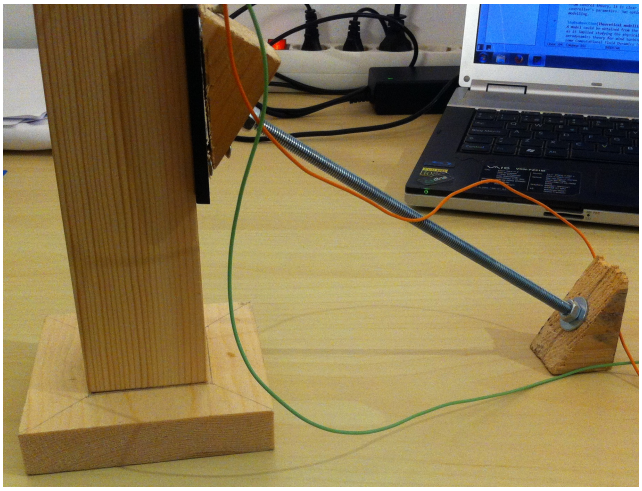
(a) Auxiliary structure attachments. (b) Fan's motor attachments.

Figure 1: Attachments of the auxiliary structure and fan's motor.

1.1.2 Electronics connections

The connections necessary to drive the fan's motor are listed below. It is advised to follow the same order as proposed. This avoids any risk of short-circuit or other undesired counter-effect. The final result can be observed in Figure 3.

1. Connect the motor's terminal to the motor driver [3] output power labelled as "Motor A". Use a screwdriver to fasten the connections.
2. Connect the motor driver to the power supply [4]. "VMS" labelled connection corresponds to the positive output of the power supply while "GND" corresponds to the negative output of the power supply.



(a) Fan's tower auxiliary structure.

(b) Gimlet used to make the screw holes.

Figure 2: Auxiliary structure and gimlet.

3. Connect the micro-controller (Arduino UNO rev3) to the motor driver. Link Arduino's pin 9 with motor's driver "ENA" input pin (note that not all Arduino pins can output a PWM signal [5]). This is the entry to modulate motor's speed. Then, two Arduino's digital pins (from 2 to 13, it is advised not to use pins 0 and 1 as they can not be used for digital I/O when serial communication with the computer is used) should be connected to the motor's driver input pins "IN1" and "IN2" [6]. These two pins control the direction of motion. Depending on how the motor's terminals have been connected to the driver, if "IN1" is at high state and "IN2" is at a low state, the motor will spin clockwise or counter-clockwise. However, if the input pins' state is reversed, the direction of rotation will do so.
4. Connect the Arduino to the computer through the USB wire. This is necessary to load on the micro-controller the program it must execute. This connection is also responsible for providing power to the Arduino. Moreover, some information could be shown or entered through the Arduino's serial monitor.
5. Before connecting the power supply to the electric grid, the Arduino program should be loaded on it. If not, the previous sketch loaded on Arduino will be executed. Some undesired consequence could happen as it could do anything unexpected. Then, plug the power supply to the electric grid.

1.1.3 Arduino sketch

```
int ENA=9;
int IN1=2;
int IN2=3;
int mv=128;

void setup(){
  pinMode(ENA,OUTPUT);
  pinMode(IN1,OUTPUT);
  pinMode(IN2,OUTPUT);
```

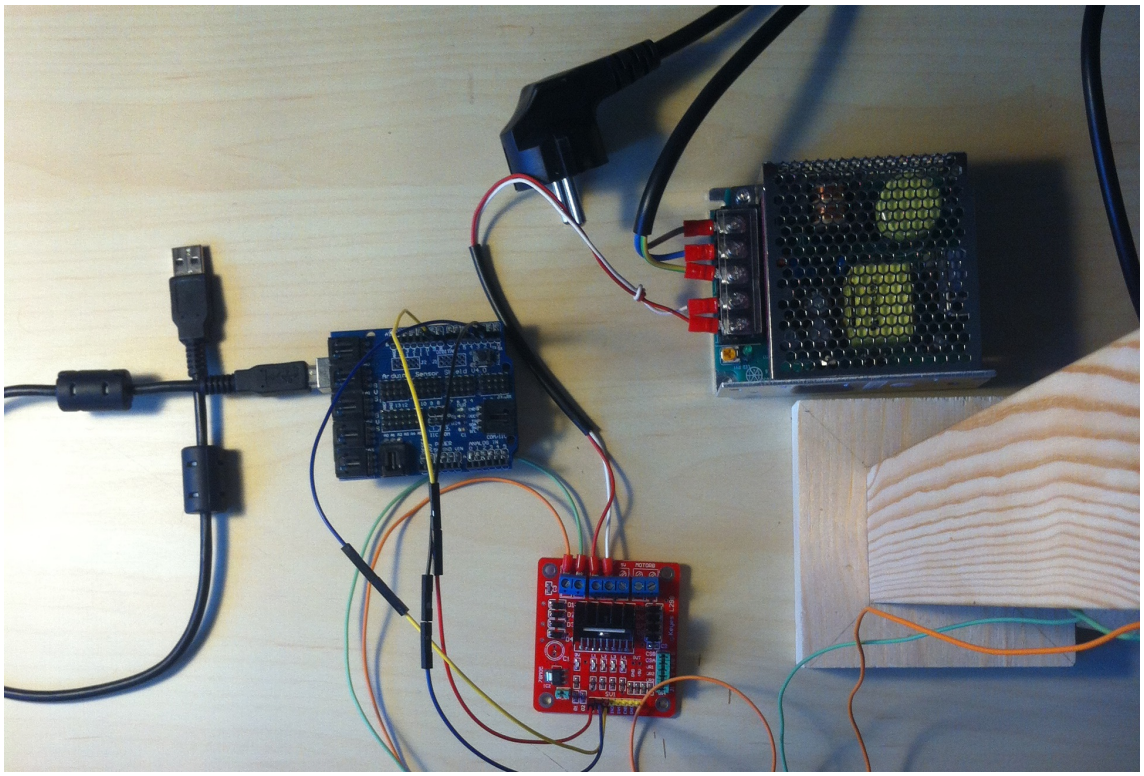


Figure 3: Layout of the connections required to drive the fan's motor.

```
    digitalWrite(ENA,LOW);
    digitalWrite(IN1,HIGH);
    digitalWrite(IN2,LOW);
}

void loop(){
    analogWrite(ENA,mv);
}
```

The code is structured as follows.

1. Variables declaration. Each pin is labelled with the driver's pin it should be connected to. In this manner, if it is necessary to change the Arduino pin, only one line needs to be modified. Note that mv stands for manipulated variable.
2. Set-up. The pins are defined as outputs and a direction of motion is set. Even though, the motor will not spin at this stage as the PWM output is constantly at low state.
3. Loop. The motor's speed is modulated thanks to the PWM output. Remind that it is an 8 bit value, so it can vary from 0 to 255.

1.2 Wind turbine's rotary encoder

The rotary encoder can be tested before including it into the wind turbine's structure. So, first, the connections to read its measurements will be explained. Then, the code used

with Arduino will be described. And, finally, the instructions to incorporate it to the wind turbine will be given.

1.2.1 Electronics connections

The following procedure is proposed to connect the rotary encoder to the micro-controller.

1. Solder a header to the encoder's wires. This is necessary to connect them afterwards to the Arduino pins. It is not necessary to solder the orange and uncovered wires as they will not be used. The orange wire outputs a pulse when a complete turn has been made. Usually, the wind turbine's hub will not rotate so much. The uncovered wire should be used if some important electromagnetic interference would be present.
2. Link the two encoder's channels outputs at the pins 2 and 3. It is important that these pins are used because an interrupt function will be used to read the encoder. These two pins are the only ones to where an interrupt can be attached if the Arduino UNO rev3 is used. Moreover, it is proposed to use channel A (black wire) for pin 2 and channel B (white wire) for pin 3.
3. Connect the blue wire to the Arduino's ground pin "GND" and the brown wire to the 5V pin "5V". Note that any other digital pin (apart from 0 to 3 because they will be already in use) could have been used for this purpose. The only extra requirement is to set the digital I/O pin as an output and set their state as high (for the brown wire) or low (for the blue wire).

1.2.2 Arduino sketch

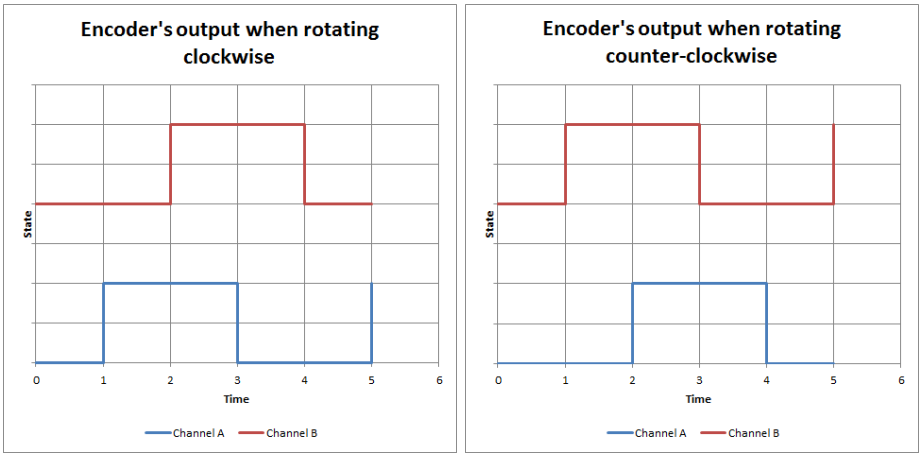
```
#include "Arduino.h"
#include <digitalWriteFast.h>
#define NOT_AN_INTERRUPT -1 //to avoid errors in compilation (from:
    https://forum.arduino.cc/index.php?topic=353602.0)

#define pinA 2
#define pinB 3

volatile long ticks=0;
volatile bool Astate;
volatile bool Astate_ant=0;
volatile bool Bstate;
volatile bool Bstate_ant=0;
volatile long time=0;

void setup()
{
   Serial.begin(115200);
   pinMode(pinA, INPUT);
   digitalWrite(pinA, LOW); //activate pullup resistors
   pinMode(pinB, INPUT); //activate pullup resistors
   digitalWrite(pinB, LOW);
   attachInterrupt(digitalPinToInterrupt(pinA), readEncoder, CHANGE);
   attachInterrupt(digitalPinToInterrupt(pinB), readEncoder, CHANGE);
}
```

```
void loop()
{
time=micros();
   while(1){
      if(micros()>time+1000000){
      time=micros();
      Serial.print(micros()); Serial.print(";");
      Serial.println(ticks);
      }
   }
}

void readEncoder(){
   Astate=digitalReadFast(pinA);
   Bstate=digitalReadFast(pinB);
   ticks=ticks+updateReading();
   Astate_ant=Astate;
   Bstate_ant=Bstate;
}

int updateReading(){
   if(Astate_ant==0 && Bstate_ant==0){
      if(Astate==1 && Bstate==0) return -1;
      if(Astate==0 && Bstate==1) return 1;
   }else if(Astate_ant==1 && Bstate_ant==0){
      if(Astate==1 && Bstate==1) return -1;
      if(Astate==0 && Bstate==0) return 1;
   }else if(Astate_ant==1 && Bstate_ant==1){
      if(Astate==0 && Bstate==1) return -1;
      if(Astate==1 && Bstate==0) return 1;
   }else if(Astate_ant==0 && Bstate_ant==1){
      if(Astate==0 && Bstate==0) return -1;
      if(Astate==1 && Bstate==1) return 1;
}

}
```

The code is structured as follows.

1. The libraries needed to execute the sketch are added. Note that digitalReadFast is used to increase the micro-controller's speed when reading the digital pins.
2. Variables definition.
   - The pins 2 and 3 are defined as constants in order to gain the speed provided by the digitalReadFast library. If not, the speed would be the same as the usual digitalRead [7].
   - Variable ticks is increased by 4096 at each revolution. It is defined as long to track the angular position even if the hub has rotated a lot.
   - Variable time is defined as long in order to measure time with microseconds to gain precision.
   - Most variables are defined as volatile so that they can be modified in an interrupt service routine [8].

3. Set-up
    - The serial connection between the Arduino and the PC is set at the highest possible baud rate in order to gain speed when printing the measurements. In this manner, the sample time is conditioned at a minimum.
    - The pins are set as inputs. If the 5V and GND pins were not used, the respective digital pins used to replace their functions should be defined as outputs here. Their state should be established as well.
    - The interrupt functions are attached to the pins 2 and 3.
4. Loop.
    - The time is read.
    - An inner loop is used to avoid using the delay function which can produce some bad results.
    - When the sampling time has passed, the measurements are printed through the Arduino serial monitor.
5. Read encoder.
    - The current channel's state is read.
    - The rotated angle is updated.
    - The old channel's state is updated for the next reading.
6. Update reading. Depending on the previous and current states of the channels, the rotated angle is updated. Figure 4 can help to understand how the function determines the output.



(a) Encoder's output when rotating clockwise.

(b) Encoder's output when rotating counter-clockwise.

Figure 4: Encoder's possible outputs depending on the sense of rotation.

1.2.3 Assembly

To accomplish this task, some instructions are proposed below. The final result of the encoder and motor assembly is shown in Figure 5.

1. Structure.
    - Drill 4 holes of 6 mm diameter for the threaded bars that will fasten the rotary encoder and hold the gear motor. The holes should be on the section's square diagonals and 28 cm from the center. This provides sufficient space to, afterwards, place the rotary encoder.
    - Screw the threaded bars in the holes. Their length should be about 17 cm. Then, fasten them to the wood with one nut for each bar.
2. Rotary encoder.
    - The wood plate that will be used to fasten the rotary encoder to the structure was built with a laser cutter. However, it could also be built with a saw, drill and rasp. Nevertheless, the precision is always better the laser cutter.
    - The encoder should be attached to the wood plate with 3 M3 screws.
3. The encoder is placed at the top of the wood tower. Then, the wood plate is fastened to the threaded bars with one nut for each bar.
4. A pinion gear should be joint to the encoder's shaft to link it with the gear motor.

1.3 Wind turbine's motor

To ensure the gear motor used to orientate the wind turbine's hub, it is first required to introduce it to the wind turbine's tower. Then, the electronic connections should be made. Finally, the Arduino sketch should be loaded to the micro-controller.

1.3.1 Assembly

To accomplish this task, some instructions are proposed below.

1. Gear motor.
    - Place the motor inside its enclosure.
    - The wood plates that will fasten the motor should be built with a laser cutter or using more tradition techniques.
    - Attach the enclosure to the wood plate that will join it to the structure. 2 M3 bolts should be used for this.
    - Join the motor's shaft to a shaft coupler using its set-screw.
2. Start to build the transmission bar. Its final result can be seen in Figure 6.
    - Attach a pinion gear to a 9 cm long plain bar. This gear will be in contact with the encoder's one.
    - Insert the transmission bar through the motor's wood plate hole built for this purpose.
3. Fasten the motor to the structure.
    - Add one nut and one washer to each threaded bar.

Figure 5: Result of the wind turbine's rotary encoder and gear motor assembly.

- Introduce the 4 threaded into the motor's wood plate holes made for this purpose.
- Add one nut and one washer to each threaded bar to fasten the wood plate to the structure.
- Insert one nut and one washer to each bar to hold another wood plate.
- Add this second wood plate to keep the motor's shaft vertical.
- Include a washer and a nut to each bar to fasten this second wood plate.
- Attach a pinion gear to the top of the transmission bar. It will link the motor's rotation with that of the transmission bar.

4. Attach wind turbine's hub to the 6 mm diameter plain bar and pinion gear. The final result can be observed in Figure 7.
   - Make a hole of 6 mm diameter to introduce a 5 cm long plain bar of 6 mm diameter.
   - Mark the 4 holes for the motor hub and make them with a gimlet.
   - Attach the motor hub to the wind turbine's hub with 4 M3 bolts.
   - Join the plain bar to the motor's hub with the set-screw. Do the same with the pinion gear.
5. Link the wind turbine's hub to the gear motor using the shaft coupler.

1.3.2 Electronics connections

As it is not necessary to read the angular position for this test, the connections that should be made are similar to the ones explained in Section 1.1.2. However, there are some differences.

- The gear motor terminals should be connected to the other output of the driver, labelled as "MOTOR B".
- The driver should be connected to the Arduino using the other pins (that were unused before): "ENB", "IN3" and "IN4".
- For the PWM output pin which will be connected to "ENB", pins 3, 5, 6, 9, 10, and 11 can be used [5].
- For the pins "IN3" and "IN4", any of the digital pins could be used except of the PWM output pin, as it will be already used.

1.3.3 Arduino sketch

Because it is not required to read the encoder's measurements, the sketch used for the fan's motor (see Section 1.1.3) can be used again. However, the pins name should be adapted to the connections made for this test.

Figure 6: Note the transmission bar used to link gear motor's movement with that of the encoder.

Figure 7: Hub's attachment to the 6 mm diameter plain bar and pinion gear.

1.4 Wind vane

To make sure that the wind's direction is appropriately measured, the wind vane tower has to be assembled. After that, the electronics components should be connected. Finally, the Arduino code has to be written down.

1.4.1 Assembly

The following instructions are offered to assemble all the wind vane's components. The final result is shown in Figure 8.

1. Add 4 threaded bars (7 cm long and 6 mm diameter) to the wood tower. Their holes should be on the section square's diagonal and at 28 cm from its center. It is suggested to use a drill to make them.
2. Build the wind vane using paper board.
    - The vertical tail is 34 cm per 14 cm.
    - The longitudinal structure is 50 cm long. A hole has to be made at 20 cm from one extreme. It is advised to use two or three layers of paper board and stick them with white glue to increase its resistance.
    - To couple the vertical tail with the longitudinal structure, cut out a rectangle of the same height of the longitudinal structure's width and half the length of the vertical tail from both elements. Then, join them. Note that the rectangle should be removed from the closest extreme to the hole of the longitudinal structure.

- Pass a 6 mm diameter and 5 cm long threaded bar through the longitudinal structure's hole. Use a pair of washers and nuts to fasten the threaded bar to the longitudinal structure.

3. Attach the encoder to its wood plate.
   - The wood plate to join the encoder to the tower's threaded bars must be built. It is advised to make it using a laser cutter. However, more traditional techniques can be used.
   - Use 3 M3 bolts to attach the encoder to the wood plate.
4. Integrate the wind vane to the encoder's shaft thanks to a coupler.
5. Fasten the encoder and wind vane to the threaded bars using nuts and washers.
6. Add a weight to balance the wind vane. The mass to use depends on distance from encoder's axis. It is suggested to pick up a small weight and find the distance to where it should be placed by trying different positions until the static equilibrium is reached.

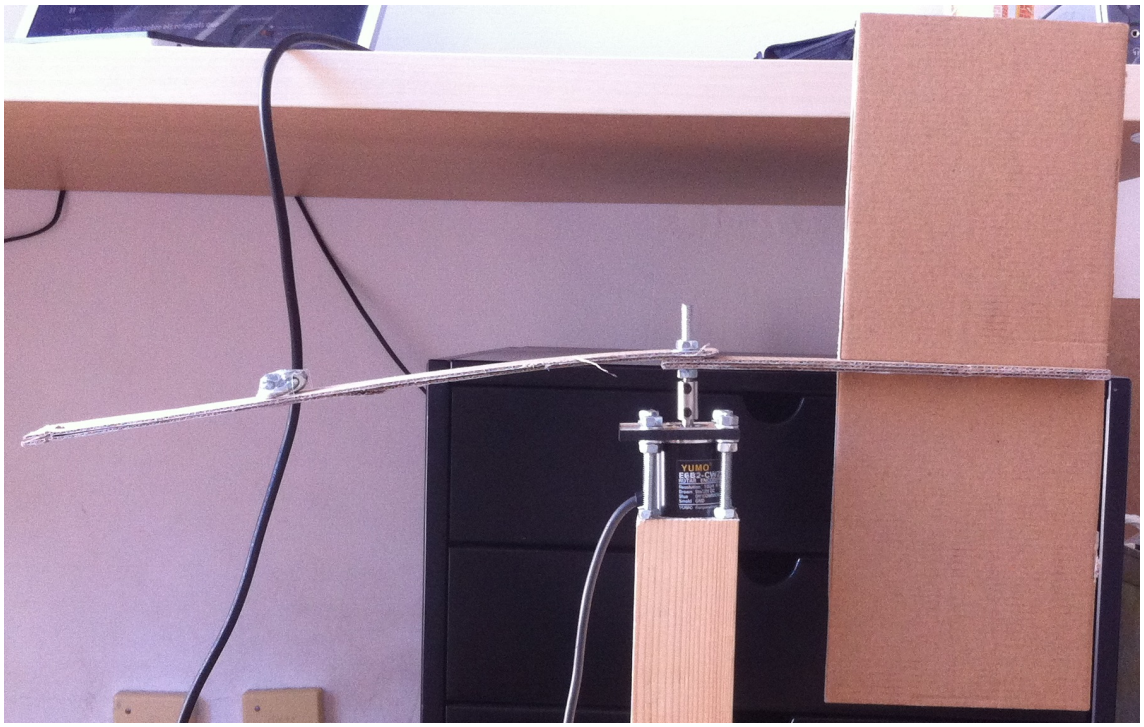


Figure 8: Final result of wind vane's assembly.

1.4.2 Electronics Connections

The connections required to carry out this test are the same used to check the fan's motor (see Section 1.1.2) and the wind turbine's rotary encoder (see Section 1.2.1).

1.4.3 Arduino sketch

```
#include "Arduino.h"
#include <digitalWriteFast.h>
#define NOT_AN_INTERRUPT -1 //to avoid errors in compilation (from:
    https://forum.arduino.cc/index.php?topic=353602.0)

#define pinA 2
#define pinB 3

volatile long ticks=0;
volatile bool Astate;
volatile bool Astate_ant=0;
volatile bool Bstate;
volatile bool Bstate_ant=0;

int start=0;
long t, tant, tstart;
int tsamp=10000;
int ENA=9;
int IN1=6;
int IN2=7;

void setup()
{
   Serial.begin(115200);
   pinMode(pinA, INPUT);
   digitalWrite(pinA, LOW);
   pinMode(pinB, INPUT);
   digitalWrite(pinB, LOW);
   attachInterrupt(digitalPinToInterrupt(pinA), readEncoder, CHANGE);
   attachInterrupt(digitalPinToInterrupt(pinB), readEncoder, CHANGE);
   pinMode(IN1,OUTPUT);
   pinMode(IN2,OUTPUT);
   pinMode(ENA,OUTPUT);
   digitalWrite(IN1,HIGH);
   digitalWrite(IN2,LOW);
   digitalWrite(ENA,LOW);
   Serial.println("Enter 'Y' when you want to start");
}

void loop(){
   if(Serial.available()>0){
      char answer=Serial.read();
      if(answer=='Y'){
         start=1;
         analogWrite(ENA,255);
         tstart=micros();
      }else{
         start=0;
         analogWrite(ENA,0);
      }
   }
   while(start==1){
      t=micros();
      if(t>=tant+tsamp){
         tant=micros();
         Serial.print(micros()); Serial.print(";");
```

```
            Serial.println(ticks);
        }
        if(t>=tstart+30000000){
            start=0;
            analogWrite(ENA,0);
        }
    }
}

void readEncoder(){
    Astate=digitalReadFast(pinA);
    Bstate=digitalReadFast(pinB);
    ticks=ticks+updateReading();
    Astate_ant=Astate;
    Bstate_ant=Bstate;
}

int updateReading(){
    if(Astate_ant==0 && Bstate_ant==0){
        if(Astate==1 && Bstate==0) return -1;
        if(Astate==0 && Bstate==1) return 1;
    }else if(Astate_ant==1 && Bstate_ant==0){
        if(Astate==1 && Bstate==1) return -1;
        if(Astate==0 && Bstate==0) return 1;
    }else if(Astate_ant==1 && Bstate_ant==1){
        if(Astate==0 && Bstate==1) return -1;
        if(Astate==1 && Bstate==0) return 1;
    }else if(Astate_ant==0 && Bstate_ant==1){
        if(Astate==0 && Bstate==0) return -1;
        if(Astate==1 && Bstate==1) return 1;
    }
}
```

The code is structured very similarly to the one used to read the wind turbine's rotary encoder (see Section 1.2.2). However some changes have been made.

- The program requires a message from the user to start. In this case, the user must send the 'Y' character through the serial monitor.
- When the start message is received, the fan's motor starts to spin generating the wind.
- The user has 30 seconds to move the fan in order to check whether the wind direction is being correctly measured. After this time, the fan's motor will stop and no more measurements from the encoder will be print through the serial monitor.
- If the user desires to start again the test, the serial monitor should be reopened and the 'Y' character sent.

2 Modelling experiments

The modelling experiments were necessary to determine the sampling time and to characterise the plant. Moreover, they were also useful to ensure Arduino was measuring and filtering the speed correctly. This section has been ordered as follows.

- Speed Modelling: obtain measurements of the wind turbine's hub angular position. The speed is calculated afterwards with Excel.
- Calculate Speed: it is checked that Arduino obtains similar results than Excel when calculating the angular speed.
- Filter Speed: Arduino results when filtering the speed measurements are compared to the ones obtained when using Excel.

2.1 Speed Modelling

The purpose of this code is to print encoder's measurements at the specified sample time. The test's main characteristics are listed below.

- The motor is always driven by the same PWM signal. For the sketch shown, it is set to 70 but any value from 0 to 255 can be used. Recall that the PWM signal is an 8 bit value.
- Speed is not measured by Arduino who simply prints encoder's readings through the serial monitor. As it is not required to perform many tasks, the micro-controller can operate at very small sample times.
- It is necessary to send a specific message through the serial monitor for the program to start. After a certain time, the gear motor will stop spinning.
- The electronics connections are the ones used in Sections 1.2.1 and 1.3.2.

```
int start=0;
unsigned long t;
unsigned long tant;
int tsamp=10000;
unsigned long tend;

int ENB=10;
int IN3=4;
int IN4=5;

#include "Arduino.h"
#include <digitalWriteFast.h>
#define NOT_AN_INTERRUPT -1 //to avoid errors in compilation (from:
    https://forum.arduino.cc/index.php?topic=353602.0)

#define pinA 2 //black
#define pinB 3 //white
// brown 5V and blue GND

volatile long ticks=0;
volatile bool Astate;
volatile bool Astate_ant=0;
volatile bool Bstate;
volatile bool Bstate_ant=0;

void setup(){
   Serial.begin(115200);
   Serial.println("Press Y when you want to start: ");
```

```
   pinMode(ENB,OUTPUT);
   pinMode(IN3,OUTPUT);
   pinMode(IN4,OUTPUT);
   digitalWrite(ENB,LOW);
   digitalWrite(IN3,HIGH);
   digitalWrite(IN4,LOW);

   pinMode(pinA, INPUT);
   digitalWrite(pinA, LOW);
   pinMode(pinB, INPUT);
   digitalWrite(pinB, LOW);
   attachInterrupt(digitalPinToInterrupt(pinA), readEncoder, CHANGE);
   attachInterrupt(digitalPinToInterrupt(pinB), readEncoder, CHANGE);
}

void loop(){
   if(Serial.available()>0){
      char answer=Serial.read();
      if(answer=='Y'){
         start=1;
         tend=micros()+5000000;
      }
   }
   while(start==1){
      t=micros();
      if(t-tant>tsamp){
         tant=t;
         analogWrite(ENB,70);
         Serial.print(micros()); Serial.print(";");
         Serial.println(ticks);
         if(micros()>tend){
            start=0;
            analogWrite(ENB,0);
         }
      }
   }
}

void readEncoder(){
   Astate=digitalReadFast(pinA);
   Bstate=digitalReadFast(pinB);
   ticks=ticks+updateReading();
   Astate_ant=Astate;
   Bstate_ant=Bstate;
}

int updateReading(){
   if(Astate_ant==0 && Bstate_ant==0){
      if(Astate==1 && Bstate==0) return -1;
      if(Astate==0 && Bstate==1) return 1;
   }else if(Astate_ant==1 && Bstate_ant==0){
      if(Astate==1 && Bstate==1) return -1;
      if(Astate==0 && Bstate==0) return 1;
   }else if(Astate_ant==1 && Bstate_ant==1){
      if(Astate==0 && Bstate==1) return -1;
      if(Astate==1 && Bstate==0) return 1;
   }else if(Astate_ant==0 && Bstate_ant==1){
```

```
      if(Astate==0 && Bstate==0) return -1;
      if(Astate==1 && Bstate==1) return 1;
   }
}
```

2.2 Calculate Speed

This test aims to check that the Arduino is calculating the speed appropriately. The microcontroller's results are compared to the ones obtained with Excel. It they are not satisfactory enough, the factor (112) used to determine angular speed from two time adjacent position readings, can be modified until better results are gained. It is structured in a very similar manner to the code used to model the motor's speed behaviour (see Section 2.1). Note that the electronics connections are the same than in the aforementioned test.

```
int start=0;
unsigned long t;
unsigned long tant;
int tsamp=10000;
unsigned long tend;

int ENB=10;
int IN3=4;
int IN4=5;

#include "Arduino.h"
#include <digitalWriteFast.h>
#define NOT_AN_INTERRUPT -1 //to avoid errors in compilation (from:
    https://forum.arduino.cc/index.php?topic=353602.0)

#define pinA 2 //black
#define pinB 3 //white
// brown 5V and blue GND

volatile long ticks=0;
volatile bool Astate;
volatile bool Astate_ant=0;
volatile bool Bstate;
volatile bool Bstate_ant=0;

long ticks_ant=0;
int wm=0;

void setup(){
   Serial.begin(115200);
   Serial.println("Press Y when you want to start: ");

   pinMode(ENB,OUTPUT);
   pinMode(IN3,OUTPUT);
   pinMode(IN4,OUTPUT);
   digitalWrite(ENB,LOW);
   digitalWrite(IN3,HIGH);
   digitalWrite(IN4,LOW);

   pinMode(pinA, INPUT);
```

```
   digitalWrite(pinA, LOW);
   pinMode(pinB, INPUT);
   digitalWrite(pinB, LOW);
   attachInterrupt(digitalPinToInterrupt(pinA), readEncoder, CHANGE);
   attachInterrupt(digitalPinToInterrupt(pinB), readEncoder, CHANGE);
}

void loop(){
   if(Serial.available()>0){
      char answer=Serial.read();
      if(answer=='Y'){
      start=1;
      tend=micros()+5000000;
      analogWrite(ENB,128);
      }
   }
   while(start==1){
      t=micros();
      if(t-tant>tsamp){
         tant=t;
         wm=(ticks-ticks_ant)*117;
         analogWrite(ENB,128);
         Serial.print(micros()); Serial.print(";");
         Serial.print(ticks); Serial.print(";");
         Serial.println(wm);
         ticks_ant=ticks;
         if(micros()>tend){
            start=0;
            analogWrite(ENB,0);
         }
      }
   }
}

void readEncoder(){
   Astate=digitalReadFast(pinA);
   Bstate=digitalReadFast(pinB);
   ticks=ticks+updateReading();
   Astate_ant=Astate;
   Bstate_ant=Bstate;
}

int updateReading(){
   if(Astate_ant==0 && Bstate_ant==0){
      if(Astate==1 && Bstate==0) return -1;
      if(Astate==0 && Bstate==1) return 1;
   }else if(Astate_ant==1 && Bstate_ant==0){
      if(Astate==1 && Bstate==1) return -1;
      if(Astate==0 && Bstate==0) return 1;
   }else if(Astate_ant==1 && Bstate_ant==1){
      if(Astate==0 && Bstate==1) return -1;
      if(Astate==1 && Bstate==0) return 1;
   }else if(Astate_ant==0 && Bstate_ant==1){
      if(Astate==0 && Bstate==0) return -1;
      if(Astate==1 && Bstate==1) return 1;
   }
}
```

2.3 Filter Speed

After ensuring that the micro-controller can calculate the angular speed appropriately, it is time to check it can filter it to remove some of the caught noise. The electronics connections are the same when modelling the angular speed or when ensuring that it was correctly calculated (see Sections 2.1 and 2.2). Moreover, the code structure is very similar. However, some lines have been added to filter the speed measurement. For information about the filter discrete equation used in the code, it is suggested to consult the project's Report.

```
int start=0;
unsigned long t;
unsigned long tant;
int tsamp=10000;
unsigned long tend;

int ENB=10;
int IN3=4;
int IN4=5;

#include "Arduino.h"
#include <digitalWriteFast.h>
#define NOT_AN_INTERRUPT -1 //to avoid errors in compilation (from:
    https://forum.arduino.cc/index.php?topic=353602.0)

#define pinA 2 //black
#define pinB 3 //white
// brown 5V and blue GND

volatile long ticks=0;
volatile bool Astate;
volatile bool Astate_ant=0;
volatile bool Bstate;
volatile bool Bstate_ant=0;

long ticks_ant=0;
int wm=0;
long wf=0, wf_ant=0;

void setup(){
   Serial.begin(115200);
   Serial.println("Press Y when you want to start: ");

   pinMode(ENB,OUTPUT);
   pinMode(IN3,OUTPUT);
   pinMode(IN4,OUTPUT);
   digitalWrite(ENB,LOW);
   digitalWrite(IN3,HIGH);
   digitalWrite(IN4,LOW);

   pinMode(pinA, INPUT);
   digitalWrite(pinA, LOW);
   pinMode(pinB, INPUT);
   digitalWrite(pinB, LOW);
   attachInterrupt(digitalPinToInterrupt(pinA), readEncoder, CHANGE);
   attachInterrupt(digitalPinToInterrupt(pinB), readEncoder, CHANGE);
```

```
}

void loop(){
   if(Serial.available()>0){
      char answer=Serial.read();
      if(answer=='Y'){
         start=1;
         tend=micros()+5000000;
         analogWrite(ENB,128);
      }
   }
   while(start==1){
      t=micros();
      if(t-tant>tsamp){
         tant=t;
         wm=(ticks-ticks_ant)*117;
         wf=(5*wf_ant+wm)/6; //filter set at Tc=0,05s
         analogWrite(ENB,128);
         Serial.print(micros()); Serial.print(";");
         Serial.print(ticks); Serial.print(";");
         Serial.print(wm); Serial.print(";");
         Serial.println(wf);
         ticks_ant=ticks;
         wf_ant=wf;
         if(micros()>tend){
            start=0;
            analogWrite(ENB,0);
         }
      }
   }
}

void readEncoder(){
   Astate=digitalReadFast(pinA);
   Bstate=digitalReadFast(pinB);
   ticks=ticks+updateReading();
   Astate_ant=Astate;
   Bstate_ant=Bstate;
}

int updateReading(){
   if(Astate_ant==0 && Bstate_ant==0){
      if(Astate==1 && Bstate==0) return -1;
      if(Astate==0 && Bstate==1) return 1;
   }else if(Astate_ant==1 && Bstate_ant==0){
      if(Astate==1 && Bstate==1) return -1;
      if(Astate==0 && Bstate==0) return 1;
   }else if(Astate_ant==1 && Bstate_ant==1){
      if(Astate==0 && Bstate==1) return -1;
      if(Astate==1 && Bstate==0) return 1;
   }else if(Astate_ant==0 && Bstate_ant==1){
      if(Astate==0 && Bstate==0) return -1;
      if(Astate==1 && Bstate==1) return 1;
   }
}
```

3 Control algorithms tests

As the controller is in a cascade form, the different loops that are included in it are tested. In this manner, the final controller was free of errors as all its loops were tested previously in an isolated way.

- Speed controller: the analytically tuned I+P controller was tested.
- Ziegler-Nichols: the I+P speed controller's set point is established proportionally to the error. The first value of the proportional constant that make the system oscillate in the steady state is sought.
- Position controller: Using Ziegler-Nichols rules a proportional position controller was tuned. It was tested to ensure that the results obtained were satisfactory.
- Windy conditions: after obtaining acceptable behaviour when there is no wind, it is important to check that the system is able to work under windy conditions.

3.1 Speed controller

The connections required to conduct this test are the same ones used to model the plant (see Section 2). The code is also based on the ones developed when modelling the system. However, in this case, the manipulated variable is calculated at each sample time taking into account the encoder's measurement and the speed set point.

Note that the controller's parameters (K_P and K_I) have been introduced in their fractional form. This is because the Arduino works better with integers than with floating numbers [9]. For more information on how the controller's discrete equation was obtained, it is suggested to consult the project's Report.

```
int start=0;
unsigned long t;
unsigned long tant;
int tsamp=10000;
unsigned long tend;

int ENB=10;
int IN3=4;
int IN4=5;

#include "Arduino.h"
#include <digitalWriteFast.h>
#define NOT_AN_INTERRUPT -1 //to avoid errors in compilation (from:
    https://forum.arduino.cc/index.php?topic=353602.0)

#define pinA 2 //black
#define pinB 3 //white
// brown 5V and blue GND

volatile long ticks=0;
volatile bool Astate;
volatile bool Astate_ant=0;
volatile bool Bstate;
volatile bool Bstate_ant=0;
```

```
long ticks_ant=0;
int wm=0;
long wf=0, wf_ant=0;
int sp=4096;
long mv=0, mv_ant=0;

void setup(){
   Serial.begin(115200);
   Serial.println("Press Y when you want to start: ");

   pinMode(ENB,OUTPUT);
   pinMode(IN3,OUTPUT);
   pinMode(IN4,OUTPUT);
   digitalWrite(ENB,LOW);
   digitalWrite(IN3,HIGH);
   digitalWrite(IN4,LOW);

   pinMode(pinA, INPUT);
   digitalWrite(pinA, LOW);
   pinMode(pinB, INPUT);
   digitalWrite(pinB, LOW);
   attachInterrupt(digitalPinToInterrupt(pinA), readEncoder, CHANGE);
   attachInterrupt(digitalPinToInterrupt(pinB), readEncoder, CHANGE);
}

void loop(){
   if(Serial.available()>0){
      char answer=Serial.read();
      if(answer=='Y'){
         start=1;
         tend=micros()+5000000;
      }
   }
   while(start==1){
      t=micros();
      if(t-tant>tsamp){
         tant=t;
         wm=(ticks-ticks_ant)*117;
         wf=(5*wf_ant+wm)/6; //filter set at Tc=0,05s
         mv=mv_ant+21*(wf_ant-wf)/674+(sp-wf)/571;
         if(mv>255) mv=255;
         if(mv<-255) mv=-255;
         if(mv>0){
            digitalWrite(IN3,HIGH);
            digitalWrite(IN4,LOW);
            analogWrite(ENB,mv);
         }else{
            digitalWrite(IN3,LOW);
            digitalWrite(IN4,HIGH);
            analogWrite(ENB,-mv);
         }
         Serial.print(micros()); Serial.print(";");
         Serial.print(ticks); Serial.print(";");
         Serial.print(wm); Serial.print(";");
         Serial.print(wf); Serial.print(";");
         Serial.println(mv);
         ticks_ant=ticks;
```

```
            wf_ant=wf;
            mv_ant=mv;
            if(micros()>tend){
               start=0;
               analogWrite(ENB,0);
            }
         }
      }
   }

   void readEncoder(){
      Astate=digitalReadFast(pinA);
      Bstate=digitalReadFast(pinB);
      ticks=ticks+updateReading();
      Astate_ant=Astate;
      Bstate_ant=Bstate;
   }

   int updateReading(){
      if(Astate_ant==0 && Bstate_ant==0){
         if(Astate==1 && Bstate==0) return -1;
         if(Astate==0 && Bstate==1) return 1;
      }else if(Astate_ant==1 && Bstate_ant==0){
         if(Astate==1 && Bstate==1) return -1;
         if(Astate==0 && Bstate==0) return 1;
      }else if(Astate_ant==1 && Bstate_ant==1){
         if(Astate==0 && Bstate==1) return -1;
         if(Astate==1 && Bstate==0) return 1;
      }else if(Astate_ant==0 && Bstate_ant==1){
         if(Astate==0 && Bstate==0) return -1;
         if(Astate==1 && Bstate==1) return 1;
      }
   }
```

3.2 Ziegler-Nichols

After ensuring the speed controller worked well, the position controller was tuned using Ziegler-Nichols rules. This required to use a proportional loop over the speed loop. Then, the loop's gain was increased until the orientation oscillated in the steady state. The connections used are exactly the same than for the speed controller test (see Section 3.1). The only difference in the code with respect to the speed controller is that the speed set point is determined proportionally to the error between the measured position and the position set point.

```
int start=0;
unsigned long t;
unsigned long tant;
int tsamp=10000;
unsigned long tend;

int ENB=10;
int IN3=4;
int IN4=5;
```

```
#include "Arduino.h"
#include <digitalWriteFast.h>
#define NOT_AN_INTERRUPT -1 //to avoid errors in compilation (from:
    https://forum.arduino.cc/index.php?topic=353602.0)

#define pinA 2 //black
#define pinB 3 //white
// brown 5V and blue GND

volatile long ticks=0;
volatile bool Astate;
volatile bool Astate_ant=0;
volatile bool Bstate;
volatile bool Bstate_ant=0;

long ticks_ant=0;
int wm=0;
long wf=0, wf_ant=0;
long sp_w=0, sp_p=512;
long mv=0, mv_ant=0;

void setup(){
   Serial.begin(115200);
   Serial.println("Press Y when you want to start: ");

   pinMode(ENB,OUTPUT);
   pinMode(IN3,OUTPUT);
   pinMode(IN4,OUTPUT);
   digitalWrite(ENB,LOW);
   digitalWrite(IN3,HIGH);
   digitalWrite(IN4,LOW);

   pinMode(pinA, INPUT);
   digitalWrite(pinA, LOW);
   pinMode(pinB, INPUT);
   digitalWrite(pinB, LOW);
   attachInterrupt(digitalPinToInterrupt(pinA), readEncoder, CHANGE);
   attachInterrupt(digitalPinToInterrupt(pinB), readEncoder, CHANGE);
}

void loop(){
   if(Serial.available()>0){
      char answer=Serial.read();
      if(answer=='Y'){
         start=1;
         tend=micros()+5000000;
      }
   }
   while(start==1){
      t=micros();
      if(t-tant>tsamp){
         tant=t;
         wm=(ticks-ticks_ant)*117;
         wf=(5*wf_ant+wm)/6; //filter set at Tc=0,05s
         sp_w=24*(sp_p-ticks)/10;
         mv=mv_ant+21*(wf_ant-wf)/674+(sp_w-wf)/571;
         if(mv>255) mv=255;
```

```
        if(mv<-255) mv=-255;
        if(mv>0){
           digitalWrite(IN3,HIGH);
           digitalWrite(IN4,LOW);
           analogWrite(ENB,mv);
        }else{
           digitalWrite(IN3,LOW);
           digitalWrite(IN4,HIGH);
           analogWrite(ENB,-mv);
        }
        Serial.print(micros()); Serial.print(";");
        Serial.print(sp_p); Serial.print(";");
        Serial.print(ticks); Serial.print(";");
        Serial.print(wm); Serial.print(";");
        Serial.print(wf); Serial.print(";");
        Serial.println(mv);
        ticks_ant=ticks;
        wf_ant=wf;
        mv_ant=mv;
        if(micros()>tend){
           start=0;
           analogWrite(ENB,0);
        }
     }
  }
}

void readEncoder(){
  Astate=digitalReadFast(pinA);
  Bstate=digitalReadFast(pinB);
  ticks=ticks+updateReading();
  Astate_ant=Astate;
  Bstate_ant=Bstate;
}

int updateReading(){
  if(Astate_ant==0 && Bstate_ant==0){
     if(Astate==1 && Bstate==0) return -1;
     if(Astate==0 && Bstate==1) return 1;
  }else if(Astate_ant==1 && Bstate_ant==0){
     if(Astate==1 && Bstate==1) return -1;
     if(Astate==0 && Bstate==0) return 1;
  }else if(Astate_ant==1 && Bstate_ant==1){
     if(Astate==0 && Bstate==1) return -1;
     if(Astate==1 && Bstate==0) return 1;
  }else if(Astate_ant==0 && Bstate_ant==1){
     if(Astate==0 && Bstate==0) return -1;
     if(Astate==1 && Bstate==1) return 1;
  }
}
```

3.3 Position controller

After determining the critical gain for the system (see Section 3.2), Ziegler-Nichols rules allowed to calculate the position controller's parameters. The proportional controller seemed sufficient. The code and the connections used to check the position controller's response are the same that were used to determine the system's critical response (see Section 3.2). Note that, in this case, the proportional gain to determine speed set point from position error is the one calculated thanks to Ziegler-Nichols rules.

3.4 Performance under windy conditions

The final test to ensure that the project's requirements were fulfilled is to try the controller using the wind to set the desired position. The electronics connections are the ones used to check the wind vane's correct measurements (see Section 1.4.2) and the speed controller (see Section 3.1). A general view of the experimental set used for this test can be observed in Figure 9.

However, the code used is a little bit different from the previous ones. As it is required to use 2 different encoders and the Arduino only has 2 pins where an interrupt function can be attached, the way to measure the angular position was modified. See project's Report for more information on this issue.

Recall that the encoder's precision was halved, so the speed controller's parameters had to be divided by two as well. To understand why the precision decreases, it is advised to look at the differences between the functions used to read the encoder for this test and for the previous ones along with Figure 4. It was not necessary to modify the position loop parameters because the static gain between position and speed is unitary.

Moreover, the position set point does not equal the wind vane's measurements. Sometimes, the wind vane can oscillate due to turbulence phenomenons. To avoid the wind turbine to follow these oscillations, the position set point is only changed when the measured wind direction changes more than 5º with respect to the current position set point.

Finally, it is advised to first perform the test without wind and moving the wind vane's with the hand. In this way, the controller can be tried in a safer way. Then, the fan could be activate to do the final test. To turn on or off the fan, it is only necessary to change the PWM sent to the motor driver with the analogWrite command. If its argument is 0, the PWM signal will have no duty cycle, so the fan's motor will not spin. If the analogWrite argument is 255, the PWM sent will have a 100% duty cycle that will make the fan's motor spin at full speed.

```
int start=0;
unsigned long t;
unsigned long tant;
int tsamp=10000;
unsigned long tend;

int ENA=9;
int IN1=6;
int IN2=7;

int ENB=10;
int IN3=11;
```

```
int IN4=12;

#include "Arduino.h"
#include <digitalWriteFast.h>
#define NOT_AN_INTERRUPT -1 //to avoid errors in compilation (from:
    https://forum.arduino.cc/index.php?topic=353602.0)

#define pinA_Motor 2 //black
#define pinB_Motor 4 //white
#define pinA_Vane 3 //black
#define pinB_Vane 5 //white
// brown 5V and blue GND

volatile long ticksM=0;
volatile bool AM;
volatile bool BM;

volatile long ticksV=0;
volatile bool AV;
volatile bool BV;

long ticksM_ant=0;
int wm=0;
long wf=0, wf_ant=0;
long sp_w=0, sp_p=0;
long mv=0, mv_ant=0;

void setup(){
   Serial.begin(115200);
   Serial.println("Press Y when you want to start: ");

   pinMode(ENA,OUTPUT);
   pinMode(IN1,OUTPUT);
   pinMode(IN2,OUTPUT);
   pinMode(ENB,OUTPUT);
   pinMode(IN3,OUTPUT);
   pinMode(IN4,OUTPUT);
   digitalWrite(ENA,LOW);
   digitalWrite(IN1,HIGH);
   digitalWrite(IN2,LOW);
   digitalWrite(ENB,LOW);
   digitalWrite(IN3,HIGH);
   digitalWrite(IN4,LOW);

   pinMode(13,OUTPUT);
   digitalWrite(13,HIGH);
   pinMode(pinA_Motor, INPUT);
   digitalWrite(pinA_Motor, LOW);
   pinMode(pinB_Motor, INPUT);
   digitalWrite(pinB_Motor, LOW);
   pinMode(pinA_Vane, INPUT);
   digitalWrite(pinA_Vane, LOW);
   pinMode(pinB_Vane, INPUT);
   digitalWrite(pinB_Vane, LOW);
   attachInterrupt(digitalPinToInterrupt(pinA_Motor), readMotor, CHANGE);
   attachInterrupt(digitalPinToInterrupt(pinA_Vane), readVane, CHANGE);
}
```

```
void loop(){
   if(Serial.available()>0){
      char answer=Serial.read();
      if(answer=='Y'){
         start=1;
         tend=micros()+10000000;
         analogWrite(ENA,255);
      }
   }
   while(start==1){
      t=micros();
      if(t-tant>tsamp){
         tant=t;
         wm=(ticks-ticks_ant)*117;
         wf=(5*wf_ant+wm)/6; //filter set at Tc=0,05s
         if(ticksV>sp_p+13 || ticksV<sp_p-13) sp_p=ticksV;
         sp_w=25*(sp_p-ticksM)/10;
         mv=mv_ant+21*(wf_ant-wf)/337+3*(sp_w-wf)/856;
         if(mv>255) mv=255;
         if(mv<-255) mv=-255;
         if(mv>0){
            digitalWrite(IN3,HIGH);
            digitalWrite(IN4,LOW);
            analogWrite(ENB,mv);
         }else{
            digitalWrite(IN3,LOW);
            digitalWrite(IN4,HIGH);
            analogWrite(ENB,-mv);
         }
         Serial.print(micros()); Serial.print(";");
         Serial.print(ticksV); Serial.print(";");
         Serial.print(ticksM); Serial.print(";");
         Serial.println(mv);
         ticksM_ant=ticksM;
         wf_ant=wf;
         mv_ant=mv;
         if(micros()>tend){
            start=0;
            analogWrite(ENA,0);
            analogWrite(ENB,0);
         }
      }
   }
}

void readMotor(){
   AM=digitalReadFast(pinA_Motor);
   BM=digitalReadFast(pinB_Motor);
   ticksM=ticksM+updateReading(AM,BM);
}

void readVane(){
   AV=digitalReadFast(pinA_Vane);
   BV=digitalReadFast(pinB_Vane);
   ticksV=ticksV+updateReading(AV,BV);
}
```

```
int updateReading(bool Astate, bool Bstate){
   if(Astate==0 && Bstate==0) return 1;
   if(Astate==0 && Bstate==1) return -1;
   if(Astate==1 && Bstate==0) return -1;
   if(Astate==1 && Bstate==1) return 1;
}
```

Figure 9: Experimental set used to check that the wind turbine could follow the wind's direction.

REFERENCES

References

[1] R. Components, "Motor de dc Como Drills 719RE360, 6 - 15 V dc, 10668 rpm, 5,75 W, 0,77 A." http://es.rs-online.com/web/p/motores-dc/2389715/.

[2] HobbyKing, "GWS Style Slowfly Propeller 14x4.7 Grey (CW) (2pcs)." http://www.hobbyking.com/hobbyking/store/__28289__GWS_Style_Slowfly_Propeller_14x4_7_Grey_CW_2pcs_.html.

[3] HobbyKing, "Kingduino Compatible H-Bridge Motor Driver." http://www.hobbyking.com/hobbyking/store/__37421__Kingduino_Compatible_H_Bridge_Motor_Driver.html.

[4] R. Components, "Fuente de alimentación conmutada integrada (SMPS), 50W, 1 salida, Tensión 12V dc, Corriente 4.2A." http://es.rs-online.com/web/p/fuentes-de-alimentacion-de-modo-conmutado-smps-integradas/7703298/.

[5] Arduino, "Arduino - AnalogWrite." https://www.arduino.cc/en/Reference/AnalogWrite.

[6] Arduino, "Introduction to the Arduino Board." https://www.arduino.cc/en/Reference/Board.

[7] D. R. Hessmer, "Quadrature Encoder too Fast for Arduino (with Solution)." http://www.hessmer.org/blog/2011/01/30/quadrature-encoder-too-fast-for-arduino-with-solution/.

[8] Arduino, "Arduino - AttachInterrupt." https://www.arduino.cc/en/Reference/AttachInterrupt.

[9] Arduino, "Arduino - Float." https://www.arduino.cc/en/Reference/Float.